REMARKS

This communication responds to the Office Action mailed on November 21, 2005. Claims 1, 9, 14, and 19 are amended, no claims are canceled, and no claims are added. As a result, claims 1-23 are now pending in this Application.

Declaration

The summary page of the instant Office Action indicates that the Office has objected to the Declaration, and that the objection is explained in the "attached Office Action or Form PTO-152. However, no explanation could be found, and the Applicant is unaware of any reason that an objection would be raised to the Declaration. Accordingly, the Applicant respectfully requests clarification regarding this objection.

§112 Rejections of the Claims

Claim 21 was rejected under 35 USC § 112, first paragraph, as failing to comply with the enablement requirement. In particular, it is asserted that "the claimed 'device option memory' is unclear." The Applicant again respectfully disagrees.

As noted in the previous response, the device option memory is well-known to those of skill in the art. In response to the Applicant's assertion, the Office states that the "definition defines an 'Option ROM', the word 'device' does not appear in the term or its definition." Perhaps a few more specific examples from industry will be persuasive.

As noted in the article "The PC-AT Boot Process and Option ROMs", pg. 4, 1999, attached hereto as Appendix A, device option memories, such as "IPL device option ROMs", can be a chief component in the personal computer (PC) boot process. One example is found in section 6.8 of the "BIOS Boot Specification", Compaq Computer Corporation, pg. 28, 1995, attached hereto as Appendix B, where it is noted that a "legacy device's option ROM" can be set up to capture interrupts. Other examples of such usage are available upon request.

Thus, the statement by the Office is not sufficient to establish a *prima facie* case of non-enablement under § 112. As noted in the previous response, the *prima facie* case requires each of the following six elements to be established:

1. a rational basis as to
    a. why the disclosure does not teach, or
    b. why to doubt the objective truth of the statements in the disclosure that purport to teach;
2. the manner and process of making and using the invention;
3. that correspond in scope to the claimed invention;
4. to one of ordinary skill in the pertinent technology;
5. without undue experimentation; and
6. dealing with subject matter that would not already be known to the skilled person as of the filing date of the application.

"The Examiner must provide evidence ... supporting *each of these elements* for a rejection under the first paragraph of § 112 to be proper." See *Patent Prosecution, Practice and Procedure Before The United States Patent Office*, Ira H. Donner, pg. 691, 2002. (emphasis added)

Since evidence supporting *each of the required elements* noted above (e.g., that one of ordinary skill would be unable to practice embodiments of the invention without undue experimentation) has not been presented, a *prima facie* case for non-compliance under § 112, first paragraph, has not been established with respect to claim 21. Reconsideration and withdrawal are respectfully requested.

Claims 1-23 were rejected under 35 USC § 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which Applicant regards as the invention. Nothing has been stated by the Office respecting the claim language in relation to how it would be interpreted by one of skill in the art (e.g., why one of ordinary skill in the art would have any difficulty implementing a "memory cached by a non-volatile cache" or an "operating system cache driver"). Indeed, examples of such terminology, as commonly used in the industry, including the Microsoft® developer network web site, have been presented in the prior office action response filed by the Applicant. Therefore, since a *prima facie* case of indefiniteness has not been established, the Applicant respectfully traverses this rejection.

However, in the interest of temporal economy, so as not to unduly prolong an interpretive difference between the Office and the Applicant, and not for reasons related to patentability, the

Applicant has amended the independent claims to specify that information in the memory is cached (e.g., "memory having information cached by the non-volatile cache"). For this reason, and because a *prima facie* case of indefiniteness has not been established under 35 USC § 112, second paragraph, it is respectfully requested that this rejection of claims 1-23 be reconsidered and withdrawn.

§103 Rejections of the Claims

Claims 1-4, 6-10, 14, 16 and 19 were rejected under 35 USC § 103(a) as being unpatentable over Sarkozy (U.S. 5,732,238; hereinafter "Sarkozy") further in view of Handy (The Cache Memory Book, Academic Press, 1998; hereinafter "Handy"). Claims 5, 17, 18 and 20 were rejected under 35 USC § 103(a) as being unpatentable over Sarkozy in view of Handy and further in view of Lee et al. (U.S. 5,937,433; hereinafter "Lee"). Claims 11 and 23 were rejected under 35 USC § 103(a) as being unpatentable over Sarkozy in view of Handy and further in view of Howard (U.S. 6,629,198; hereinafter "Howard"). Claim 12 was rejected under 35 USC § 103(a) as being unpatentable over Sarkozy in view of Handy and further in view of Heemels (U.S. 5,603,331; hereinafter "Heemels"). Claim 15 was rejected under 35 USC § 103(a) as being unpatentable over Sarkozy in view of Handy and further in view of Kozierok (PC Guide; hereinafter "PC Guide"). Claim 22 was also rejected under 35 USC § 103(a) as being unpatentable over Sarkozy in view of Handy and Lee and further in view of PC Guide. First, the Applicant does not admit that Sarkozy, Handy, Lee, Howard, Heemels, or PC Guide are prior art and reserves the right to swear behind these references in the future. Second, since a *prima facie* case of obviousness has not been established as required by M.P.E.P. § 2142, the Applicant respectfully traverses these rejections.

The Examiner has the burden under 35 U.S.C. § 103 to establish a *prima facie* case of obviousness. *In re Fine*, 837 F.2d 1071, 1074, 5 U.S.P.Q.2d (BNA) 1596, 1598 (Fed. Cir. 1988). In combining prior art references to construct a *prima facie* case, the Examiner must show some objective teaching in the prior art or some knowledge generally available to one of ordinary skill in the art that would lead an individual to combine the relevant teaching of the references. *Id.* The M.P.E.P. contains explicit direction to the Examiner that agrees with the *In re Fine* court:

> In order for the Examiner to establish a *prima facie* case of obviousness, three base criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, and not based on applicants disclosure. *M.P.E.P.* 2142 (citing *In re Vaeck*, 947 F.2d 488, 20 U.S.P.Q.2d (BNA) 1438 (Fed. Cir. 1991)).

An invention can be obvious even though the suggestion to combine prior art teachings is not found in a specific reference. *In re Oetiker*, 977 F.2d 1443, 24 U.S.P.Q.2d (BNA) 1443 (Fed. Cir. 1992). However, while it is not necessary that the cited references or prior art specifically suggest making the combination, there must be some teaching somewhere which provides the suggestion or motivation to combine prior art teachings and applies that combination to solve the same or similar problem which the claimed invention addresses. One of ordinary skill in the art will be presumed to know of any such teaching. (See, e.g., *In re Nilssen*, 851 F.2d 1401, 1403, 7 U.S.P.Q.2d 1500, 1502 (Fed. Cir. 1988) and *In re Wood*, 599 F.2d 1032, 1037, 202 U.S.P.Q. 171, 174 (C.C.P.A. 1979)). The requirement of a suggestion or motivation to combine references in a *prima facie* case of obviousness is emphasized in the Federal Circuit opinion, *In re Sang Su Lee*, 277 F.3d 1338; 61 U.S.P.Q.2D 1430 (Fed. Cir. 2002), which notes that the motivation must be supported by evidence in the record.

No proper *prima facie* case of obviousness has been established because (1) combining the references does not teach all of the limitations set forth in the claims, (2) there is no motivation to combine the references, and (3) combining the references provides no reasonable expectation of success. Each of these points will be explained in detail, as follows.

Combining The References Does Not Teach All Claim Limitations.

First, with respect to independent claims 1, 9, 14, and 19, it is admitted in the Office Action that Sarkozy does not disclose "that recording an address of a write operation should be done prior to executing an operating system driver." The Office goes on to assert that "Handy explains that disk caches are often implemented in dynamic RAM using software control." However, this fact alone does not imply any particular actions with respect to the sequencing of

disk cache control operations. In fact, the Applicant's representative was unable to find anything within the bounds of Handy to support recording write operation addresses prior to "executing an operating system cache driver," as claimed in independent claims 1, 9, 14, and 19 (claims 14 and 19 have been amended to change the term "booting" to "executing" for consistency with claims 1 and 9, and not for reasons related to patentability). Lee, Howard, Heemels, and PC Guide also fail in this respect. While the Office asserts that "write addresses can be written at any time and the cache driver can be one of many Operating System cache drivers loaded in the system," the existence of such a situation using the cited references has not been demonstrated.

Therefore, no combination of Sarkozy, Handy, Lee, Howard, Heemels, or PC Guide can provide recording write operation addresses prior to "executing an operating system cache driver", as claimed by the Applicant in independent claims 1, 9, 14, and 19, and a *prima facie* case of obviousness has not been established. Further, it is respectfully noted that if an independent claim is nonobvious under 35 USC § 103, then any claim depending therefrom is also nonobvious. *See* M.P.E.P. § 2143.03. Therefore, claims 2-8, 10-13, 15-18, and 20-23 are also nonobvious.

There Is No Motivation to Combine the References.

Handy's express concern with cache operational speed (e.g., "This book pertains only to CPU caches and not to disk caches. ... CPU caches operate at such high speed that hardware control must be used, and the cache itself must be implemented in static RAM.) teaches away from using the non-volatile cache technology of Sarkozy. *See* Handy, pg. xv. Adding any one or more of Lee, Howard, Heemels, and PC Guide does nothing to negate the teaching of Handy.

References must be considered in their entirety, including parts that teach away from the claims. See MPEP § 2141.02. The fact that references can be combined or modified does not render the resultant combination obvious unless the prior art also suggests the desirability of the combination. *In re Mills*, 16 USPQ2d 1430 (Fed. Cir. 1990); M.P.E.P. § 2143.01.

In this case, Handy teaches that such a combination would *not* be desirable. While the Office asserts that "all caches share many desirable qualities such as speed, data coherence, and fault tolerance," this assertion does nothing to overcome the express teachings of Handy. The use of unsupported assertions in the Office Action does not satisfy the explicit requirements

needed to demonstrate motivation as set forth by the *In re Sang Su Lee* court. Therefore, the Examiner appears to be using personal knowledge, and is respectfully requested to submit an affidavit as required by 37 C.F.R. § 1.104(d)(2).

Combining the References Provides No Reasonable Expectation of Success.

Implementing a disk cache "in dynamic RAM (DRAM) using software control" as taught by Handy does nothing to promote recording write operation addresses prior to "executing an operating system cache driver", as claimed by the Applicant. Since the Office admits this element is also missing from Sarkovy, combining Handy and Sarkovy provides no reasonable expectation of success with respect to the claimed order of operation. Lee, Howard, Heemels, and PC Guide also fail to add anything to this combination that leads to a reasonable expectation of success with respect to achieving the claimed order of operation.

The Office asserts that "one cannot show nonobviousness by attacking references individually." However, it is respectfully noted that it is the *combination* of Sarkovy and Handy (and/or Lee, Howard, Heemels, and PC Guide) that fails to bring about a reasonable expectation of success. This is because the required elements are missing from all of the references, and therefore, no *combination* of these references would lead to a reasonable expectation of success.

In summary, none of the references teach recording write operation addresses prior to "executing an operating system cache driver," as set forth in independent claims 1, 9, 14, and 19. No evidence has been entered in the record to support a need to combine the references (in fact the references teach away from the proposed combinations), and no reasonable expectation of success results from any combination. The requirements of *M.P.E.P.* § 2142 have not been satisfied, and a *prima facie* case of obviousness has not been established with respect to these independent claims. All dependent claims are also nonobvious, since claims depending from nonobvious independent claims are also nonobvious. It is therefore respectfully requested that the rejections to claims 1-12, 14-20, and 22-23 under 35 U.S.C. § 103 be reconsidered and withdrawn.

Allowable Subject Matter

Claim 13 was indicated to be allowable if rewritten to overcome the rejection(s) under 35 USC § 112, second paragraph, set forth in the Office Action and to include all of the limitations of the base claim and any intervening claims. However, since the Applicant believes all of the claims are in condition for allowance as currently presented, the Applicant respectfully declines to amend claim 13 at this time.

CONCLUSION

The Applicant respectfully submits that the claims are in condition for allowance and notification to that effect is earnestly requested. The Examiner is invited to telephone the Applicant's attorney Mark Muller at (210) 308-5677, or the undersigned at (612) 349-9592 to facilitate prosecution of this Application. If necessary, please charge any additional fees or credit overpayment to Deposit Account No. 19-0743.

Respectfully submitted,

ROBERT ROYER ET AL.

By their Representatives,

SCHWEGMAN, LUNDBERG, WOESSNER & KLUTH, P.A.
Attorneys for Intel Corporation
P.O. Box 2938
Minneapolis, Minnesota 55402
(612) 349-9592

Date Jan 23, 2006

By Ann M. McCrackin
Ann M. McCrackin
Reg. No. 42,858

CERTIFICATE UNDER 37 CFR 1.8: The undersigned hereby certifies that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail, in an envelope addressed to: Mail Stop AF, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on this 23rd day of January 2006.

Amy Moriarty

Name

Signature

APPENDIX A

Article: The PC-AT Boot Process and Option ROMs

Found at: http://www.intel.com/technology/magazine/communications/sv09991.pdf

The PC-AT Boot Process and Option ROMs

Roy Wade
Staff Software Engineer
Storage Components Division
LSI Logic Corporation

Ramin Neshati
Developer Guides
Technical Program Manager
Intel Corporation

Table of Contents

(Click on page number to jump to sections)

The PC-AT Boot Process and Option ROMs

Roy Wade
Staff Software Engineer
Storage Components Division
LSI Logic Corporation

Ramin Neshati
Developer Guides
Technical Program Manager
Intel Corporation

Overview

The ubiquitous PC-AT architecture has reached a point where system designers and Independent Hardware Vendors (IHVs) are actively looking for ways to extend its capabilities, performance, ease-of-use, scalability, and other design characteristics. Much has changed since the introduction of the PC in the early 1980s: the system processors, I/O bus, memory map, boot process, graphics, etc., have all undergone drastic improvements. However, we stand today at the cusp of even greater technology transitions in systems design, allowing for vastly improved system performance, compatibility, multi-vendor interoperability, and "headroom" for scalability, future expansion, and growth.

The introduction of the IA-64 processor family allows system designers to make a clean break from the past and introduce new and innovative system designs that will do for server systems what the earlier Intel x86 processors did for the desktop PC—a massive proliferation of high-performance, low-cost server systems and solutions. Attempts at reducing or removing legacy technologies, allowing room for innovation, and extending the ease-of-use, performance, and scalability characteristics of server systems based on the Intel® Architecture have received special attention recently. A group of leading companies in the computer industry has formed working groups to collectively create an evolutionary path for the transition away from legacy technologies. LSI Logic and Intel, both leading companies in the markets they serve, have been at the forefront of paving the way for extending the Intel Architecture into the enterprise and the data center.

The PC-AT boot process lies at the core of this legacy migration trend. The introduction of the Extensible Firmware Interface (EFI), an abstraction interface that allows the de-coupling of the hardware from the operating system boot loader and provides other run-time services, has been enthusiastically adopted by the industry. However, the initial release of the EFI specification has not fully addressed an area that is of special importance to IHVs: option ROMs. However, efforts are already afoot to address this issue in the next release of the EFI specification. Quite simply, an option ROM allows the IHV to supply additional firmware to boot and configure its peripheral device and to extend the functionality of the system firmware (BIOS) during the boot process. There is no shortage of documentation detailing the PC-AT boot sequence. As PC technology has evolved, the number of specifications covering this topic has grown resulting in a fragmented image that takes some effort to grasp.

This article provides a comprehensive discussion of the PC-AT boot process in terms of today's technologies and architectures. Specifically discussed are PCI expansion ROMs for Plug and Play (PnP) devices and the details of their role in the boot process. A companion white paper that describes a legacy-reduced system boot process is currently in the early stages of formulation. We will defer the discussion of EFI option ROMs to this near-future effort.

Option ROMs Defined

The following is a general description of the current IA-32 option ROM architecture. Only Plug and Play compatible option ROMs are discussed. For a comprehensive and historical perspective, refer to the Plug and Play BIOS specification.

As was mentioned, an option ROM is a firmware component associated with a specific add-on device. The associated device may reside on the system-board or on a Plug and Play card. Accordingly, a device's option ROM will reside in system ROM or in PnP card ROM. Option ROMs are intended to isolate a hardware device by providing an abstract interface that implement device-specific functions, including:

- Power-on self-test
- Initialization
- Interrupt service routines
- BIOS routines

The I/O services provided by an option ROM serve as a translation layer between differing protocols. For example, an option ROM may provide a legacy Int 10H video interface to an AGP device; or provide a legacy Int 13h disk drive interface to one or more disk drives attached to a SCSI bus. The PnP option ROM model is generic and can support any Plug and Play device. Requirements specific to an application are detailed in existing specifications and are beyond the scope of this article (e.g., a PCI-to-USB PnP product would be concerned with the appropriate PCI and USB specifications).

Typically a PnP device will provide at least one option ROM image, but some architectures allow for more (e.g., PCI). It is not required that a PnP device provides an associated option ROM, hence they are optional. Option ROMs facilitate the Plug and Play concept by adhering to a common image format, which indicates flexible resource requirements in support of automatic configuration.

During the boot process, option ROM images are copied from their ROM to main memory (RAM) and their ROM address range is mapped (shadowed) to main memory. This is done to increase execution speed by taking advantage of the faster access time of RAM versus ROM. The system BIOS is responsible for this expansion process during boot. The system BIOS loads and initializes option ROM images at its discretion. In newer systems, only those option ROMs required to boot the operating system are loaded. The operating system then loads its own drivers for all PnP devices while shadowed option ROMs remain resident (and dormant) in RAM.

The primary purpose of the pre-boot subsystem is to provide a set of services to find and execute an operating system loader in a reliable and expeditious manner. The pre-boot environment can also provide error checking, error logging/reporting and error recovery as well as support system expansion.

Two chief components in the boot process are the system BIOS and Initial Program Load (IPL) device option ROMs. The system BIOS is a single instance entity that controls the sequence of boot events. It provides abstracted services, enumerates hardware devices, loads option ROMs, and performs resource management. The system BIOS gains control of the initial boot process by intercepting a power on or reset. It invokes the system bootstrap loader to load the operating system (possibly via an option ROM) from the IPL device and execute it. As discussed previously, option ROMs provide services to access the devices attached to their bus, in this case the boot device.

PC-AT Boot Model

The following discussion of the PC-AT boot model assumes that the reader is familiar with:

PnP compatibility for all components (system BIOS, devices, option ROMs, operating system)
PCI 2.1 compliance
BIOS Boot Specification support for system BIOS and IPL device option ROMs
Device Driver Initialization Model (DDIM) as specified in the PnP BIOS Specification

Figure 1 below shows an example system configuration that may prove helpful in a description of the boot sequence.

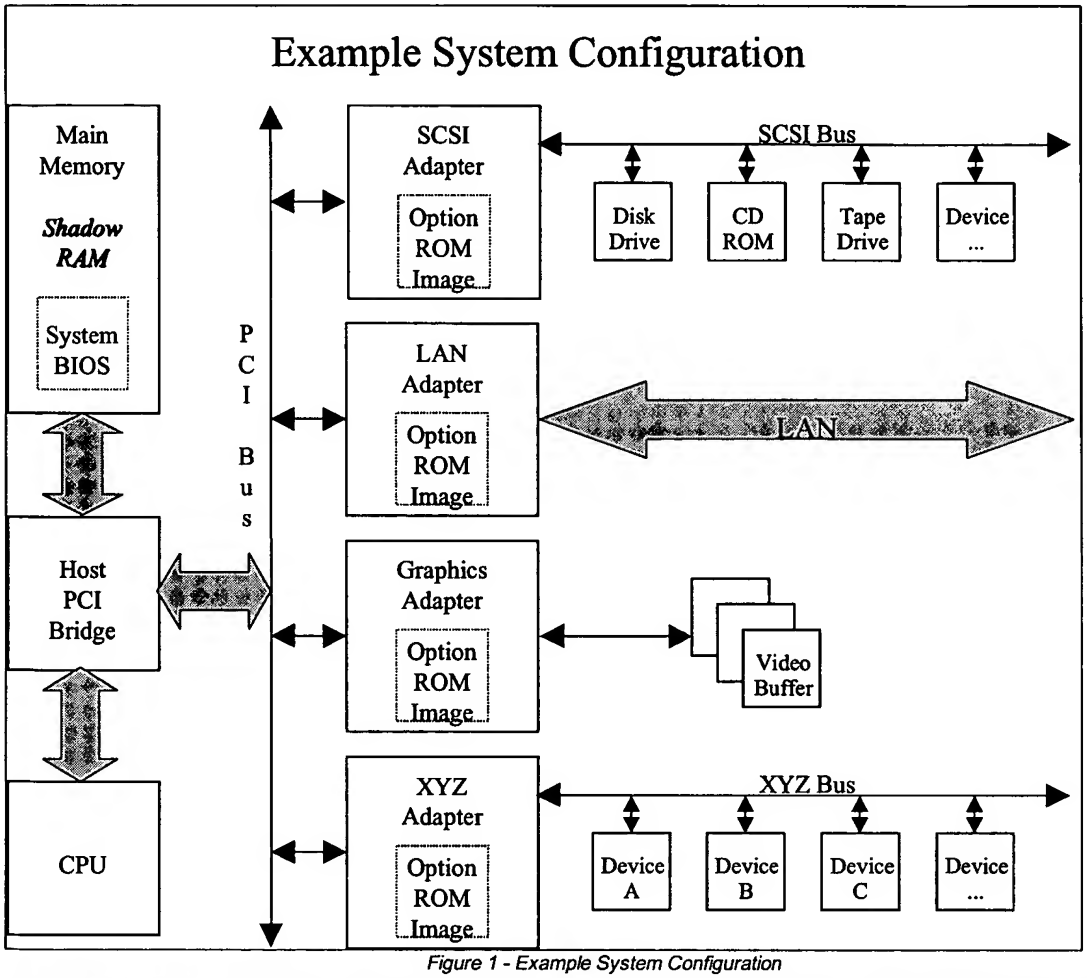


Figure 1 - Example System Configuration

The BIOS Boot Specification defines a method to specify the IPL devices to attempt to boot from and prioritize their order. This method requires that option ROMs identify their IPL devices to the system BIOS and that the system BIOS provides a mechanism for the user to control the IPL order. The comprehensive description of this mechanism can be found in the BIOS Boot (BBS) Specification and is not repeated here.

IPL Sequence of Events

Below is the sequence of events to define the IPL order. In this example, it is assumed that at least one IPL candidate device does exist.

IPL Sequence of Events:

- The system BIOS captures control of the system after power on or reset.
- The system BIOS performs the following functions:
- Initializes the primary output device (display).
- Initializes the primary input device (keyboard).
- Performs self-test functions (RAM test etc.).
- The system BIOS scans the system's busses (e.g., PCI) for PnP option ROM Headers.
- The system BIOS performs resource conflict resolution and allocation. The algorithm used in the resource allocation process is beyond the scope of this specification. In the case of PCI, the resource requirements (IRQs, I/O, memory, etc.) for an option ROM are specified in the Configuration Registers.
- The system BIOS displays a prompt that the operator uses to enter the Setup Utility.
- The system BIOS reads previous settings from non-volatile memory.
- The system BIOS scans the system's busses to detect PnP expansion headers for option ROMs

that provide access to potential IPL devices. For each IPL device option ROM that is found:
- The system BIOS copies the ROM image into main memory.
- The system BIOS calls the image's initialization entry point, which is specified in the PnP option ROM Header. The option ROM performs the following:
- Performs required initialization, and determines the device(s) it is to control.
- If there are no devices to support, the option ROM "resizes" itself to zero in on its option ROM Header, then returns from its initialization entry point.
- If there are devices to support, the option ROM creates a linked list with a new PnP expansion header for each device found. The PnP expansion header fields will uniquely identify each device. A field of particular interest to the user will be the Product Name String. The option ROM then returns from its initialization entry point.
- At this point all option ROMs with PnP expansion headers have been initialized. Option ROMs with device(s) to support will not have hooked any interrupts (e.g., Int 13h).
- The system BIOS now scans main memory to detect all PnP expansion headers for potential IPL devices. For each PnP expansion headers that is found:
- If a BEV header, the entry is added to the IPL priority list.
- If a BCV header, the entry is added to the BCV priority list.

NOTE: See [BIOS Boot Specification Version 1.01] for the distinction between the IPL and BCV priority lists.
- The system BIOS presents the IPL and BCV priority lists to the user via a Setup menu.
- The user manipulates the IPL and BCV priority lists as desired, saves the settings, then exits Setup.
- The system BIOS saves the settings to non-volatile memory and automatically reboots the system.

PnP BBS System

We are now ready to discuss the sequence of events that occur during the boot process of a PnP BBS system. The focus is directed toward the interaction between the system BIOS and option ROMs.

Plug and Play Bios Boot System Sequence of Events:

- The system BIOS captures control of the system after power on or reset.
- The system BIOS performs the following functions:
- Initializes the primary output device (display).
- Initializes the primary input device (keyboard).
- Performs self-test functions (RAM test, etc.).
- The system BIOS scans the system's busses (e.g., PCI) for PnP option ROM headers.
- The system BIOS performs resource conflict resolution and allocation. The algorithm used in the resource allocation process is beyond the scope of this specification, see [PnP BIOS specification reference]. In the case of PCI, the resource requirements (IRQs, I/O, memory, etc.) for an option ROM are specified in the Configuration Registers.
- The system BIOS displays a prompt that the operator can use to enter the Setup utility. The prompt is ignored in this case.
- The system BIOS scans the systems' busses (e.g., PCI) to detect PnP expansion headers for option ROMs that provide access to potential IPL devices. For each IPL device option ROM that is found:
- The system BIOS copies the ROM image into main memory.
- The system BIOS calls the image's initialization entry point, which is specified in the PnP option ROM header. The option ROM performs the following:
- Initialization. It also determines if the device(s) it is to control.
- If there are no devices to support, the option ROM "resizes" itself to zero in its option ROM header then returns from its initialization entry point.
- If there are devices to support, the option ROM creates a linked list with a new PnP expansion header for each device found. The PnP expansion header fields will uniquely identify each device. The option ROM then returns from its initialization entry point.

- At this point, all option ROMs with PnP expansion headers have been initialized. Option ROMs with device(s) to support will not have hooked any interrupts (e.g., Int 13h).
- The system BIOS reads settings from non-volatile memory, including the IPL and BCV priority lists. If non-volatile memory is corrupt, default IPL and BCV priority lists are created.
- The system BIOS builds the run-time BCV priority list.
- Main memory is scanned to detect all PnP expansion headers that have BCV entry points.
- Each BCV entry that is found is appended to the run-time BCV priority list.
- The system BIOS compares the NVM and run-time BCV priority lists (actually it compares the number of items in the lists). For the purposes of this discussion we assume the lists are the same (if the lists do not match the system BIOS may clear the NVM BCV priority list). Either way, the run-time BCV priority list is used from this point on.
- The system BIOS calls each BCV entry (extracted from the PnP expansion header) in the order specified in the BCV priority list. For each entry in the BCV priority list, the system BIOS calls the associated BCV entry point to request the option ROM to install Int 13h services.

The option ROM performs the following:

- The option ROM determines if it actually controls any Int 13h IPL devices.
- If there are no IPL devices, the option ROM simply returns from its BCV entry point. If there are IPL devices, the option ROM proceeds.
- The option ROM determines the number of hard drives currently installed by reading BDA address 0040:0075.
- If no other hard drives are installed (i.e., BDA 0040:0075 is zero, so this is the first hard drive) the current Int13h vector is copied to the Int 40h vector so that floppy services are handled properly.
- The option ROM infers the drive number for the current BCV IPL device from BDA 0040:0075 (i.e., BDA 0040:0075 + 80h).
- The option ROM hooks the Int13h vector, saving the current vector in order to chain to services for other drive numbers. Note that the option ROM may install Int 13h services for more than one drive in a single "hook".
- The option ROM increments BDA 0040:0075 by the number of drives it installed services for, typically the number of drives installed is one.
- The option ROM returns from its entry point.
- At this point the option ROMs for all IPL devices have been initialized, all BCV device Int 13h services have been installed, and shadow RAM has been read-only enabled.
- The system BIOS sets the Int 18h vector to the address of its failed boot attempt recovery entry. This function is responsible for walking through the IPL priority list on failed boot attempts.
- The system BIOS sets the Int 19h vector to the address of its boot strap loader. This function is responsible for loading and invoking an IPL device's bootstrap loader or BEV entry point.
- The system BIOS executes Int 19h. The first entry in the IPL priority list is the specified boot device.

Int 19h Processing:

- If the current IPL priority list selection is a BCV device:
- The system BIOS reads the device's bootstrap loader from Sector 1, Head 0, Cylinder 0 to address 0000:7C00. The Int13h services for the BCV device's associated drive number is used to read the loader.
- If a valid boot sector is detected, the system BIOS loads the OS bootstrap loader and calls its entry point. If the OS loader fails, it executes an Int 18h to indicate the failed boot. If the OS loader is successful, the OS is in control and there is no return to Int 19h.
- If the current IPL priority list selection is a BEV device:
- The system BIOS calls the BEV entry (extracted from the PnP expansion header). The BEV entry may also hook Int 13h, for example if the BEV is for an "El Torito" CD-ROM Hard Drive emulation.
- The BEV loads and executes its OS bootstrap loader.
- If the BEV or OS loader fails, it executes an Int 18h to indicate the failed boot. If the OS loader is successful, the OS is in control and there is no return to Int 19h.

- Int 18h is executed to indicate a failed boot.
- Int 18h Processing
- If all devices in the IPL priority list have been attempted:
- An error message/prompt is displayed indicating no OS was found.
- When the user responds to the prompt, the system BIOS executes Int 19h. The first entry in the IPL priority list is the specified boot device (i.e., starts over again).
- Otherwise, the system BIOS executes Int 19h, specifying the next relative IPL priority list entry as the boot device.

Single Option ROM Code Image

We now direct our attention to the description of the components and format of a single option ROM code image. Option ROMs that follow the Device Driver Initialization Model (DDIM) (PnP BIOS specification reference) exist in two forms:
Pre-initialization Image—the original option ROM image as programmed into the system-board or PnP card.
Post Initialization Image—the option ROM image as it exists in shadow RAM after its initialization entry has successfully returned.

Figures 2, and 3 show the logical components of an option ROM image which supports PnP, PCI, BBS, and DDIM. The individual elements are described below.

Figure 2: Pre-initialization Option ROM Image
Figure 3: Post-initialization Option ROM Image

Option ROM Components

PnP Option ROM Header: This is the root element of a Plug and Play option ROM. It identifies an image as an option ROM and provides links to the PCI Data Structure and PnP expansion header. The system BIOS uses the information in this header to:

- Detect a PnP option ROM image
- Copy (shadow) an image into RAM
- Validate an image via a checksum
- Initialize an image by calling its initialization entry
- Locate an image's PCI Data Structure
- Locate an image's PnP expansion header

Start with BIOS Boot Specification Version 1.01 for a detailed description.

PCI Data Structure: This is a companion item to the PnP option ROM Header. It qualifies an image as a PCI Expansion ROM and contains identification information for an image and its device.

Refer to PCI BIOS Specification Revision for a detailed description.

PnP Expansion Headers: This is a companion item to the PnP option ROM Header. In a pre-initialization image it identifies an image as a PnP option ROM. In a post-initialization image, one or more of these headers may exist, each identifying an IPL device, including:

- Reference to Product Name String
- Reference to BEV Code
- Reference to BCV Code
- Reference to the next PnP expansion header (i.e., linked list of IPL devices)

Start with BIOS Boot Specification Version 1.01 for a detailed description.

Runtime Code: The Runtime Code is the constant data and executable code that remains after an option ROM has successfully returned from its initialization entry. This code includes:

- BEV Code (BIOS Boot Specification Version 1.01)—code to directly load an OS from an IPL device and optionally hook Int 13h.
- BCV Code (BIOS Boot Specification Version 1.01)—code to detect an IPL device and optionally hook Int 13h.
- Int 13h Service Code—conventional and enhanced [EDD specification reference] Int 13h disk drive services.

Initialization Code: The Initialization Code is the constant data and executable code that is used only during initialization then discarded before returning from initialization entry. This code is responsible for:

- Device detection
- Device initialization
- Initializing runtime data structures
- Creation of Product Name String for each detected device
- Creation of PnP expansion header for each detected device
- "Disposing" of itself per the DDIM, see (Initialization Sequence paragraph reference)

Pad: Unused space used to expand an image to the next greatest 2048 byte boundary. Legacy requirements dictate that an image's size be a multiple of 2048 bytes.

Checksum: The checksum of the entire option ROM image. A value such that the 8-bit sum, using unsigned 2's complement addition ignoring overflow of all bytes including the checksum, is zero.

Product Name Strings: A Product Name String is a null terminated ASCII string that is intended to uniquely identify an IPL device. Currently only the first 32 characters are displayed and therefore significant.

Start with [BBS reference] for a detailed description.

Initialization Sequence: The Plug and Play (PnP) BIOS Specification (PnP BIOS Specification Reference) introduced the Device Driver Initialization Model for option ROMs. In this model, an option ROM image exists in two forms:

- Pre-initialization option ROM Image (Figure 1)—the image as it exists in ROM on the system board or PnP Card (Runtime Code and Initialization Code).
- Post-initialization option ROM Image (Figure 2)—the image as it exists in Shadow RAM after returning from its initialization entry (Runtime Code only).

The DDIM defines the transition from the Pre-initialization to the Post-initialization option ROM image. This transition facilitates:

- More efficient use of Shadow RAM by allowing initialization code (and data) to be discarded
- A means of dynamically building data structures and saving them as static (i.e., constant) data at boot time

Option ROM Initialization Sequence of Events:

- The system BIOS copies the ROM image into main memory.
- The system BIOS calls the image's initialization entry point, which is specified in the PnP option ROM Header. The option ROM performs the following:
- Initializes common data items.
- Searches for device(s) to control, for each device found:
- Initializes the device (reset, etc.) and its associated data structures.
- Creates a Product Name String to identify the device.

- Creates a PnP expansion header for the device and links it to the PnP expansion header List.
- If there are no devices to support:
- Writes zero to the Length field in its PnP option ROM Header.
- Returns from its initialization entry point with a status code indicating "no devices to support".
- Or optionally, if the option ROM decides to leave a static footprint (An option ROM may want to leave an image resident in Shadow RAM to provide data to a related option ROM or driver, for example NVS data or Vital Product Data):
- Copies any data it wishes to be static to the end of the runtime image (i.e., overwrites code).
- Updates the Length field (Length = Pre-initialization Length – Code Length + Static Data Length) in its PnP option ROM Header. This effectively discards the Code and reduces the image's footprint.
- Calculates the new image checksum and appends it to the new image.
- Returns from its initialization entry point with an appropriate status code.
- If there are devices to support, the option ROM will arrange and reduce its image as follows:
- Copies any data it wishes to be static to the end of the runtime image (i.e., overwrites Initialization Code). This would include the Product Name Strings and the PnP option ROM Headers.
- Updates the Length field (Length = Pre-initialization Length – Initialization Code Length + Static Data Length) in its PnP option ROM Header. This effectively discards the Initialization Code and reduces the image's footprint.
- Calculates the new image checksum and appends it to the new image.
- Returns from its initialization entry point with a status code indicating "successful initialization".

This article has attempted to document the legacy PC-AT boot process with special emphasis on the role of option ROMs. As part of the on-going industry effort to migrate away from legacy technologies and architectures, Intel intends to publish additional information that will describe a legacy-reduced IA server boot environment.

More Info

The following documents provide details on option ROMs and the PC-AT boot process, some of which are referenced in this article:

Sequence of Events:
IPL
PnP BBS System
Initialization

BIOS Boot Specification Version 1.01, Compaq Computer Corporation, Phoenix Technologies Ltd., Intel Corporation, 1996.

BIOS Enhanced Disk Drive Specification Version 3.0, Phoenix Technologies Ltd., 1998.

Clarification to Plug and Play BIOS Specification Version 1.0.

"El Torito" Bootable CD-ROM Format Specification Version 1.0, Phoenix Technologies, Ltd., IBM Corporation, 1994.

PCI BIOS Specification Revision (latest), PCI Special Interest Group, Hillsboro, OR.

Plug and Play BIOS Specification, Version 1.0A, Compaq Computer Corporation, Phoenix Technologies, Ltd., Intel Corporation, 1994, or ftp://download.intel.com/ial/wfm/bio10a.pdf.

POST Memory Manager Specification Version 1.01, Phoenix Technologies Ltd., Intel Corporation, 1997.

Author Bio

Roy Wade is a staff software engineer for LSI Logic Corporation. He is responsible for the design and development of BIOS and OS software to support LSI Logic's IO products. Roy has over 13 years of industry experience in a variety of areas, including satellite navigation, telecommunications, and Client/Server applications. Roy is a Phi Beta Kappa graduate of the University of North Dakota where he received his BS in Computer Science in 1986. Roy is a contributing member of the UNIX Developer's Interface Guide for IA-64 based servers (UDIG) working group. He has collaborated on patent applications in both the wireless communications and PC BIOS areas.

Ramin Neshati joined Intel in January 1998. Before becoming the developer guides technical program manager, he represented ESG on the Wired for Management initiative. Aside from Intel, Ramin has over 17 years of industry experience at S3 Incorporated, Dell Computer Corporation, Xerox Corporation and Link Systems Incorporated, in a variety of positions from software engineering to software architecture and engineering management. Ramin received his MBA from Pepperdine University (1993), the Master's degree in Computer Science from University of Idaho (1982) and the Bachelor's degree in Computer Science from Washington State University (1980). He has collaborated on several patent applications, published articles on networking protocols and services, and lectured at computer user's groups symposia, including SoftCon, EDGE and DECUS.

APPENDIX B

Article: BIOS Boot Specification

Found at: http://www.mit.edu/afs/sipb/contrib/doc/PnP/biosboot.ps

Compaq Computer Corporation

Phoenix Technologies Ltd.

Intel Corporation

BIOS Boot Specification

Version 1.00

October 11, 1995

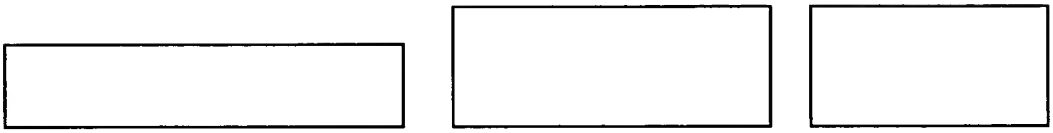


Table of Contents

1.0 Introduction

1.1 Revision History

Version	Changes
0.80	Original version.
0.81	Grammatical corrections.
1.00	Finalized for public release.

You may obtain the latest copy of the BIOS Boot Specification from the Phoenix world wide web site at http://www.ptltd.com, or by contacting a representative from one of the authoring companies.

Technical Editor:

Scott Townsend
Phoenix Technologies Ltd.
2575 McCabe Way
Irvine, CA 92714
Phone: (714) 440-8000
Fax: (714) 440-8300
Email: Scott_Townsend@ptltd.com

1.2 Related Documents

Title	Version	Author
Plug and Play BIOS Specification	1.0A	Compaq/Phoenix/Intel
Hardware Design Guide for Microsoft Windows 95	1.0	Microsoft Corporation
Enhanced Disk Drive Specification	1.1	Phoenix
"El Torito" Bootable CD-ROM Format Specification	1.0	Phoenix/IBM
PCI Local Bus Specification	2.1	PCI Special Interest Group

1.3 Purpose

The purpose of this specification is to describe a methodology by which the BIOS will identify all IPL (Initial Program Load) devices in the system, prioritize them in the order the user selects, and then sequentially go through each device and attempt to boot. The BIOS must become more intelligent about booting because the PC '95 Specification places additional requirements on the BIOS during the boot process, and there are now more devices that are bootable such as CD-ROM, network remote boot, PCMCIA, etc. It is important that this specification define a boot scheme that is generic and flexible enough to allow booting from virtually any existing IPL device, and for the definition of future IPL devices as well.

1.4 Terms

To avoid confusion and to be consistent with the Plug and Play BIOS Specification, definitions of some of the terminology used in this document are listed below.

ATA
An *Advanced Technology Attachment* drive, also known as an IDE drive, is a hard drive with the interface built-in.

ASCIIZ
An *ASCIIZ* string is simply a string of ASCII characters terminated by a NULL (0 byte).

BAID
A *BIOS Aware IPL Device* is any device that can boot an O/S, but requires the BIOS to have specific code to support it. Some examples are: the first floppy drive, the first hard drive, ATAPI CD-ROM, PCMCIA, embedded network adapter, etc.

BCV
A *Boot Connection Vector* is a pointer that points to code inside the option ROM that will perform device initialization, detect if a peripheral (such as a SCSI hard drive) is attached, and optionally hook INT 13h. The BCV resides in a PnP option ROM Expansion Header. An example of an option ROM with a BCV is a PnP ISA SCSI controller.

BDA
The *BIOS Data Area* is a data storage area in RAM. The BDA is used by the BIOS to manage the various peripherals and resources in the system. The BDA starts at segment address 0040h.

BEV
A *Bootstrap Entry Vector* is a pointer that points to code inside an option ROM that will directly load an O/S. The BEV resides in a PnP option ROM Expansion Header. An example of an option ROM with a BEV is a PnP ISA ethernet controller.

BIOS
The *Basic Input/Output System* is the software embedded on a chip located on the computer's main board. The BIOS executes POST to test and initialize the system components and then loads (boots) the O/S. The BIOS also handles the low-level input/output to the various peripheral devices connected to the computer.

Boot Device
A *Boot Device* is any device that must be initialized prior to loading the O/S. This includes the primary input device (keyboard), the primary output device (display), and the initial program load device (floppy drive, hard drive, etc.). An IPL device is one form of a boot device.

CDR
Conflict Detection and Resolution is a method by which a PnP BIOS first detects the resource requirements for PnP cards, and then allocates them in a conflict-free way.

CSN
A *Card Select Number* is a number that uniquely identifies a PnP ISA card and is used to communicate exclusively to that card. The CSN is assigned by the PnP BIOS to each PnP ISA card in the system.

DDIM
The *Device Driver Initialization Model* is a method of initializing an option ROM whereby the option ROM is first copied to shadow RAM, then its initialization vector is called with the shadow RAM write-enabled. When the option ROM completes initialization it may dispose of code not needed at run-time by re-sizing the ROM memory footprint. Finally, after the option ROM returns and the BIOS regains control, the ROM is write-protected.

DV
A *Disconnect Vector* is a pointer that points to code inside the option ROM that will perform clean-up after the Boot Connection Vector has already been called. The DV resides in a PnP option ROM Expansion Header. An example of an option ROM with a DV is a PnP ISA SCSI controller.

IPL Device
An *Initial Program Load Device* is any device in the system that can boot and load an O/S. In standard AT machines, this is the floppy drive or hard drive.

Legacy Card
A *Legacy Card* is a standard ISA card that contains no PnP compatible configurability, and no PnP Expansion Header.

NV
Non-Volatile memory is memory that is retained even when the power has been shut off. The most common type of NV memory on a PC is the CMOS RAM that is used to store system configuration information.

O/S
An *Operating System* is loaded from an IPL device when that device is selected for booting.

PFA
A *PCI Function Address* is a unique number assigned to a PCI function on a PCI device. The PFA consists of a function number, a device number, and a bus number.

PnP
Plug and Play is a term used to identify anything defined by the Plug and Play BIOS specification or the Plug and Play ISA specification. The term will typically be used to reference some device or behavior that is specific to PnP technology.

PnP Cards
PnP Cards consist of any cards that contain an option ROM with a PnP Expansion Header.

POST
The *Power-On Self Test* is the part of the BIOS that takes control immediately after the computer is turned on. POST initializes the computer hardware so that an O/S can be loaded.

Setup
The system *Setup* program is the part of the BIOS that is executed after a user specified <Hot Key> is pressed during the BIOS initialization. Setup allows the user to set up and configure the system as well as select the IPL Priority of the system.

2.0 Overview

2.1 Description

The BIOS Boot Specification defines a feature within the BIOS that creates and maintains a list of all the IPL devices found in the system and stores this list in NV memory. IPL devices come in three flavors: BAID, PnP Card, and Legacy. Only BAIDs and PnP Cards are enumerated. Legacy devices are not supported for several reasons. First, they tend to take control of the boot process altogether making them rather unfriendly. Second, they provide no means for identifying themselves as an IPL device. Finally, the BIOS cannot selectively boot from one of several Legacy IPL devices in a system.

The BIOS Boot Specification provides one basic feature, the IPL Priority. The IPL Priority is a user-specified priority of IPL devices that is arranged in Setup. This boot order is similar to the common feature of boot A: then C: or vice versa, but supports additional IPL devices. Also, the number of IPL devices in the system may vary from one power-on to another. Each time the user turns on the system all IPL devices in the system are enumerated.

Additionally, the BIOS Boot Specification defines the BCV Priority. The BCV Priority is a user-specified priority list of INT 13h Device Controllers that is arranged in Setup. This list specifies the order that the controllers will be called to install their INT 13h drive support during POST.

If an IPL device fails to load an O/S, the BIOS regains control and attempts to boot from the next available IPL device. This procedure will continue until all possible IPL devices have been exhausted. Only then will the BIOS display a message that an O/S cannot be found, wait for a key stroke, and then invoke INT 19h again. This method ensures that the BIOS has intelligently made every attempt to boot.

The BIOS Boot Specification encompasses the boot process of both PnP and non-PnP systems. The support for PnP Cards wherever mentioned is only pertinent to systems which include PnP support in their BIOS. A standard AT compatible system (also called a Legacy system) is much simpler than one with a PnP BIOS because it only supports BAIDs. A Legacy system does not need to provide any dynamic IPL device enumeration or configuration, nor does it support PnP Cards in their native mode. This is because the number of IPL devices in such a system will never change.

3.0 IPL Devices

3.1 Requirements for IPL Devices

An IPL device can be virtually any device that has the ability to load and execute an O/S. This includes floppy drives, hard drives, CD-ROM drives, PCMCIA controllers/cards, PnP Cards, Legacy cards, and could, in the future, include virtually any devices such as serial ports, parallel ports, etc. An IPL device falls into one of three categories: BAID, Legacy IPL Device, or PnP Card. There are two varieties of PnP Cards, BCV devices and BEV devices.

One of the key points of concern about an IPL device is that it be able to return back to the BIOS with an error if the O/S load fails. The BIOS Boot Specification defines INT 18h as the recovery vector for failed boot attempts. For example, if a PnP ISA network card was configured to boot from the network, but the network cable wasn't attached, the card's option ROM should signal to the BIOS that it cannot boot by invoking INT 18h.

If a Legacy card's option ROM code hooks INT 19h during its initialization call it controls the boot process. If a Legacy card hooks INT 18h, the boot failure recovery will not work. For this reason, it is recommended that all future Legacy boot devices include the PnP Expansion Header and its associated support in their option ROMs so that their priority in the booting process can be controlled.

3.1.1 IPL Table

Each BAID and BEV device must have corresponding entries in the IPL Table. An IPL Table for a typical PC would have two BAID entries, one for diskette drive A: and one for hard drive C:, and one for each BEV device found. A sample IPL Table is shown in section 4.1.

The information stored in the IPL Table consists of identification information, pointers to description strings, and pointers to handlers that will initiate an O/S load. The IPL Table and BCV Table data structure is defined in detail in Appendix A.

3.1.2 Product Name String

The Plug and Play BIOS Specification defines fields in the PnP Expansion Header for optional description strings. Vendors of PnP Cards must have meaningful strings in the Product Name String and Manufacturer String fields of the PnP Expansion Header. The Product Name String is particularly important since it will be displayed to the user as an identifier for that device. Although the only current limitation to these strings is that they be terminated by a NULL (0 byte), it is declared here that only the first 32 characters (bytes) of the Product Name String are significant. Characters beyond 32 will not be displayed. This is a reasonable length to provide the user with a unique name for that device. Product Name Strings shorter than 32 will be displayed in their entirety.

3.2 BAIDs

BAIDs include the first floppy drive, the first hard drive, and any other bootable device that has specific code in the BIOS to support it. A BAID is able to launch a bootstrap only because the BIOS has built-in support as part of the INT 19h service routine. Some examples of this might be an ATAPI compatible CD-ROM drive or a PCMCIA card/controller. BAIDs do not have option ROMs, nor can they function without specific code embedded in the BIOS.

Each BAID will have an entry in the IPL Table. This entry will contain the following information: the device type, a device sub-type, a pointer to a description string, and a pointer to code that will attempt to boot from the device. The INT 19h handler will read this table and be able to boot from any one of these devices without having to know what type of device it is. The table consists of a contiguous array of data structures of which each corresponds to a particular IPL device.

Note that only the *first* floppy drive and the *first* hard drive are considered IPL devices. The main reason why a system cannot boot from other drives is that the boot sector code specifically addresses only these drives (00h for floppy and 80h for hard drive) when INT 13h is called to load the O/S. See Appendix D for recommended changes to the O/S boot sector that could solve this problem.

3.3 Devices with PnP Expansion Headers

All IPL devices with option ROMs must contain a valid option ROM header that resides between system memory addresses C0000h and EFFFFh on a 2k boundary and begins with 55AAh. A Device's booting can only be controlled if it has a PnP Expansion Header. The Expansion Header, whose address resides within the standard option ROM header at offset +1Ah, contains important information used to configure the device. It also contains pointers to code in the device's option ROM (BCV or BEV) that the BIOS will call to boot from the device. See Appendix A for the structure of the PnP Expansion Header. There are two ways an IPL device with a PnP Expansion Header can be booted. It must contain a BCV or a BEV.

A BEV device, typically a network card, is booted by the BIOS making a far call directly to its BEV. If the device fails to boot, it executes an INT 18h which returns control back to the BIOS. BEV devices behave much like BAIDs in that they are selectively bootable.

A BCV device, typically a SCSI controller, is not directly bootable. Rather, it merely adds its drives to the system by hooking into the BIOS' INT 13h services and appending drive numbers to any existing drives. Since only drive 80h is bootable, a BCV would only be able to boot one of its drives if it installed before any other drives in the system. For this reason, it is necessary to control the order that *all* drives are installed into the system, including on-board ATA drives and those controlled by Legacy devices such as older SCSI controllers. The only way to control a BCV device's drives in the boot order is by allowing the user to specify the order of initialization among ATA, BCV, and Legacy option ROMs. This will be discussed in detail in section 5.0.

3.4 Legacy IPL Devices

This is a standard ISA card with a valid option ROM that resides in system memory address space between C0000h and EFFFFh on a 2k boundary. An example of this type of device is a Legacy SCSI hard drive controller with a bootable ROM installed. Legacy IPL devices do not contain PnP Expansion Headers in their option ROMs.

Option ROMs are detected during the BIOS' scan and their initialization vectors are called. During initialization, devices may hook INT 19h so that they will gain control when the BIOS issues INT 19h to boot the system. Or, they may hook INT 18h so that if all other IPL devices in the system fail to boot they will take over. Or, they may simply hook INT 13h and only respond to calls that reference their drive number. In any case, the first two methods provide no consistent way for the BIOS to regain control if the device fails to boot.

Legacy IPL devices will be allowed to take control of the system (via hooking interrupts) in both Legacy and PnP systems. The Plug and Play BIOS specification recommends that Legacy devices that hook a bootstrap interrupt such as INT 19h, 18h, or 13h have the interrupt re-captured by the BIOS. This is not done because grabbing an interrupt vector back after a device has hooked it can produce unpredictable results. Further, by allowing the card to take control, the behavior of these Legacy cards will be the same on both PnP and Legacy machines.

Legacy IPL devices are not enumerated or handled in any special way. Their behavior will be largely unknown by the BIOS. If a Legacy IPL device traps INT 19h it takes control of the boot process. This is acceptable because it is consistent with what Legacy systems do now. If a Legacy IPL device traps INT 13h, it becomes the local hard drive from the BIOS' perspective since it would then respond to INT 13h function calls executed in the INT 19h handler.

3.5 Identifying IPL Devices

The large variety of possible IPL devices dictates that a generic methodology be implemented so that all existing and future devices can be made bootable with a minimum of difficulty. All IPL devices in the system need to be enumerated during POST, but before INT 19h. All IPL devices fall into one of three categories: BAID, PnP Card, or Legacy.

IPL Device	Type
First floppy drive	BAID
First ATA Hard drive	BAID
PCI ATA card w/ drive	BAID
ATAPI CD-ROM drive	BAID
PCMCIA controller w/ bootable card	BAID
Ethernet controller w/ code embedded in BIOS	BAID
PnP Token Ring card	PnP (BEV)
PnP ethernet card	PnP (BEV)
Non-PnP card w/ PnP Expansion Header	PnP (BEV or BCV)
PnP SCSI card w/ drive	PnP (BCV)

3.5.1 BAIDs

All BAIDs are automatically identified because the BIOS has specific code to support them. In addition, each BAID has a corresponding entry in the IPL Table.

3.5.2 PnP Expansion Header

A PnP Card is defined by having a PnP Expansion Header in its option ROM. Any IPL device with an option ROM can contain the PnP Expansion Header, including PCI and ISA devices. If a PnP Expansion Header exists, the device will be treated as a PnP Card. If it is a BEV device it will be included in the IPL Priority. If it is a BCV device the BCV will be called. The PnP Expansion Header method of identifying boot devices with option ROMs is expected to become the future standard.

3.5.3 PCI Devices

From the standpoint of booting, most PCI IPL devices with option ROMs behave just like Legacy IPL devices. Currently, most PCI SCSI controllers will typically hook INT 13h and most PCI network cards typically hook either INT 19h or INT 18h in order to be an IPL device. It is recommended that all future PCI devices with option ROMs employ the PnP Expansion Header method for booting so that they may enjoy the benefits of appearing on a boot menu and allowing users to select its boot order.

PCI devices without option ROMs are either not an IPL device, or they behave like a BAID. For example, a PCI ATA controller will not have an option ROM, but will become the system ATA controller and utilize the BIOS' INT 13h services to interface to attached ATA drives.

The address space for option ROMs conforming to the Device Driver Initialization Model (DDIM), such as PCI, shall be write-enabled both when their initialization vector is called, and when their BCV is called.

3.5.4 Identical IPL Devices

In an implementation of the BIOS Boot Specification, the system BIOS may choose to distinguish between identical IPL devices so that the user will be able to choose the correct device, given a menu of selections. For instance, the system BIOS may recognize that two identical PnP SCSI controllers are present in the system and distinguish them by numbering them, by displaying their CSN, or some other method. Currently, no specific method is recommended nor defined here. Recognizing and dealing with identical devices is optional and not required for BIOS Boot Specification compliance.

4.0 IPL Priority

Once all BAIDs and BEV devices in the system have been identified and enumerated, the BIOS needs to decide in which order they will be selected for booting. This is handled by the IPL Priority.

The IPL Priority consists of an array of ordinal values, one for each BAID and BEV device found in the system at POST. An IPL Priority ordinal value is the index of a BAID or BEV device entry in the IPL Table. The priority for IPL devices is configurable by the user by editing the IPL Priority in Setup. The order the user selects is then stored in NV memory so that it can be retrieved by the BIOS at INT 19h time. The IPL Priority specifies not only the number of BAIDs and BEV devices in the system, but their order of booting as well.

When the INT 19h handler gains control, the first IPL device in the IPL Priority is used to attempt to boot. If that fails, the next device is tried, and so on until all devices in the IPL Priority have been tried. At that point an error message will be displayed since all attempts to boot have been exhausted.

4.1 Maintaining the IPL Priority

Since the ordinal values in the IPL Priority are simply the index of the IPL device in the IPL Table, no information is stored as to the specific IPL device that the ordinal value represents. Here we assume that the number of BAIDs in a system does not change because the BIOS contains the code for them. The BIOS recognizes only a change in the *number* of BEV devices, and not a change in the *type* of BEV devices.

If the user changes the BEV devices they have in their system without changing the total number of BEV devices, the BIOS will not detect this and will keep the previous IPL Priority. For example, if there is a PnP ISA ethernet card in a system and the user removes it and installs a different PnP ISA ethernet card, the IPL Priority would not change because the BIOS does not track individual BEV devices. It merely tracks the number of them and their order in the IPL Priority. The IPL Table would contain an entry for the new PnP ethernet card in the same location as where the old one was before. Again, as long as the total number of BEV devices doesn't change, the BIOS uses the last order that was stored in NV memory.

The way the IPL Priority is constructed is by taking the range of ordinal values for BAIDs and BEV devices, and storing them in NV memory in the IPL Priority order. The maximum number of IPL devices supported is implementation specific, and is defined at BIOS build time. The number of BAIDs also does not change. The number of BEV devices that were detected at the last boot is stored in NV memory so that a change in this number can be recognized by the BIOS. If the number of BEVs found during POST differs from the number found at last boot as indicated by the value stored in NV memory, the BIOS should either reset the IPL Priority automatically to reflect the changes, or prompt the user to enter Setup.

When the system boots for the first time, or if the NV memory is corrupted, a default IPL Priority is created. The default IPL Priority consists of the BAIDs followed by the BEV devices found in the system. The BEV devices are placed at the end of the IPL Priority in the order that they are found. Optionally, the user can be prompted to enter Setup.

The IPL Priority remains in effect until either the user changes it or the number of BEV devices in the system changes. However, the IPL Priority could change in a system with more than one BEV device if the order the BEV devices were found by the PnP BIOS changed. This seems odd at first to have the IPL Priority change and the user not be notified. But ultimately, if the user is changing boot devices they will probably want to manually edit the IPL Priority anyway. This caveat saves a lot of potential consumption of NV memory that would be used to store unique serial numbers for each BEV device. Additionally, BIOS code space is saved by not tracking individual BEV devices as opposed to simply enumerating the existing ones without reference to what type of device they are.

IPL Table

0	Floppy A:
1	Hard Drive C:
2	CD-ROM
3	BEV #1
4	BEV #2
5	
6	
7	

IPL Priority

0	1	2	3	4	5	6	7
3	**4**	**1**	**2**	**0**			

When INT 19h is executed, the BIOS will process the IPL Priority shown above in the following order: BEV #1, BEV #2, Hard Drive C:, CD-ROM, Floppy A:.

4.2 IPL Priority Pseudocode

The following pseudocode describes an example of managing the IPL Priority during POST.

Created at BIOS build time:

- First few IPL Table entries are filled in by the BAIDs.
- maxIPLCount = Number of entries in IPL Table.
- baidCount = Number of BAIDs in the system.
- maxBEVCount = (maxIPLCount - baidCount).
- The NV memory space for the IPL Priority is reserved.

Assumptions:

- The default for the IPL Priority will automatically be created during POST in case the NV memory gets corrupted.
- nvBEVCount = number of BEV devices found last time - stored in NV memory.
- postBEVCount = number of BEV devices found this time.

Execution at POST time:

- All option ROMs with a PnP Expansion Header are identified and their initialization entry points are called.
- Additional IPL Table entries are filled in with the BEV devices found.
- IF (NV memory is corrupted)
    - Set default IPL Priority by first placing the BAIDs, and then adding in the BEVs in the order they were found.
- ELSE
    - The IPL Priority is retrieved from NV memory.
    - deltaBEVCount = (nvBEVCount - postBEVCount).
    - IF (deltaBEVCount != 0)
        - IF (deltaBEVCount > 0)
            - FOR (i = 0; i != deltaBEVCount; ++i)
                - Add a new BEV device to the end of IPL Priority.
        - ELSE
            - For (i = 0; i != deltaBEVCount; --i)
                - Delete the BEV device nearest the end of IPL Priority.
        - ENDIF
        - Save the postBEVCount in NV memory as nvBEVCount.
        - Save the new IPL Priority in NV memory.
        - (Optional) - Display a message that the IPL Priority changed and allow the user to enter Setup to reconfigure the IPL Priority.
    - ENDIF
- ENDIF
- Invoke INT 19h.

5.0 BCV Priority

5.1 Introduction

In order to fully manipulate the order that IPL devices attempt to boot, it is necessary to control the order of installation of devices utilizing INT 13h. Since many INT 13h devices can be managed by one controller, we need to control the order that the controllers' INT 13h support is installed. The need to manage this ordering arises out of the currently limited booting capabilities of today's average PCs. Specifically, there is no standard way for the user to:

1. boot from a Legacy or PnP SCSI controller's hard drive when there are also ATA drives present in the system. Typically the on-board ATA support resident in the BIOS is installed before the BIOS' option ROM scan, making it the first to install into the INT 13h services. This results in the bootable drive number 80h being allocated to an ATA drive, thus making drives whose support is installed later 81h or higher and therefore non-bootable.

2. control the order that PnP BCV devices are installed into the INT 13h services. If there are several BCV devices in the system and each has a hard drive which contains a different O/S, there is no way to select booting from one device this time and a different device another time. Currently, the BCV devices are either installed randomly, or at best in the order in which they are found.

5.2 INT 13h Device Controllers

An INT 13h Device Controller installs one or more drives into the BIOS' INT 13h services by hooking the INT 13h software interrupt and chaining to the old vector. A controller can be in the form of the BIOS' own INT 13h services for ATA drives, a PnP Card with a BCV, or the collection of Legacy cards with option ROMs.

All controllers must be able to install in any order. This means before or after the BIOS installs its ATA drive support, and before or after the Legacy option ROMs are called for initialization.

5.2.1 ATA Drive Support in the BIOS

The first type of entry in the BCV Table is the BIOS' own INT 13h services for ATA drives. The ATA drive support in the BIOS must be able to install after other controllers have installed. The BIOS cannot assume that the drives it installs will occupy the first drive numbers (80h, 81h, etc.). No attempt is made to enumerate or control the installation order of individual ATA devices, although nothing defined in this specification would prevent this from being accomplished.

5.2.2 PnP Cards with BCVs

PnP Cards with BCVs are the second type of entry in the BCV Table. All PnP Cards will have their initialization vectors called before their BCV is called. Calling the BCV allows the PnP Card to install support for one or more drives into the INT 13h services.

5.2.3 Legacy Cards with Option ROMs

The third type of entry in the BCV Table is all Legacy cards with option ROMs. A Legacy card will install its INT 13h drive support when the BIOS calls the Legacy card's initialization vector. No attempt is made to enumerate or control the installation order of individual devices, although nothing defined in this specification prevents this.

5.2.4 Hard Drive BAID

The first INT 13h Device Controller that successfully installs support for drive 80h becomes the Hard Drive BAID, thus making it an IPL device and relating it to the IPL Priority.

5.2.5 Controller Installation Guidelines

- Each controller may install INT 13h support for one or more drives.
- When INT 13h is hooked, the old vector will be saved.
- A controller will only respond to requests which specify the drive number for which it has control, and will pass on requests to other drive numbers to the old vector.
- A controller may not know if it has any drives attached until it is called to install. If no drives are attached, then the controller should not hook INT 13h.
- The number of hard drives currently installed is stored in the BDA at address 0040:0075. When a controller installs support for additional drives, this location must be incremented by the number of drives that are to be added.
- A controller must install INT 13h support by using sequentially increasing drive numbers starting after the drives previously installed. For example, if two drives are already installed when the controller gets called, they will occupy drive numbers 80h and 81h. The next available numbers for the controller to occupy would be 82h, 83h, etc.
- A controller checks if the location at 0040:0075 is zero to determine if it is the first to install. If it is first, the controller must copy the INT 13h vector over the INT 40h vector so that floppy services are handled properly.
- The first controller to install will get drive number 80h. The controller then knows that it controls the hard drive boot device.
- Once a controller has hooked into INT 13h services, it can never unhook, since controllers hooked in after it would be lost.
- Any controllers such as Legacy cards which hook INT 18h or INT 19h may end up taking over the boot process and may possibly reduce the capability of the BIOS to control the boot order.

5.2.6 Notes on INT 13h Devices

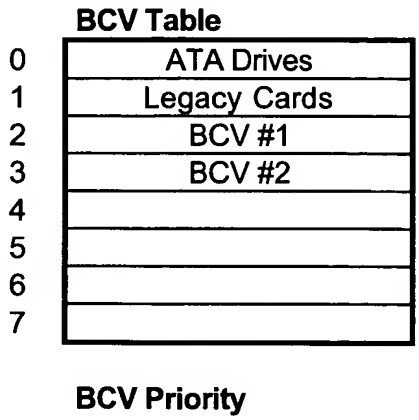
- No assumptions can be made by the controller as to what types of devices are installed in the system. For instance, if a PnP ISA SCSI card's BCV is called to install, and support for other drives is already present, the BCV handler cannot assume that the existing drives are on-board ATA drives.

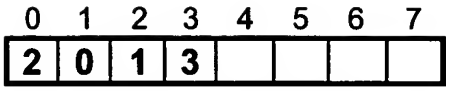
- Neither controllers nor O/S's should make assumptions about how INT 13h drive numbers are assigned. The installation ordering of INT 13h Device Controllers allows controllers to install drivers in any order selected by the user. This could create compatibility problems with some O/S's that assume drive 80h is an ATA drive. The Enhanced Disk Drive (EDD) Specification describes a method by which an ATA drive's resources may be determined.
- Drive numbers assigned to ATA and ATAPI drives for primary and secondary channels, as well as master and slave drives, are not guaranteed to be consecutive. The assignment of drive numbers is done by the INT 13h Device Controller when it installs its support. The controller may choose to assign drive numbers without regard to whether the drives are on a primary or secondary channel, or configured as master or slave.

5.3 Installation Ordering

The BIOS can control the order of installation of controllers of INT 13h devices if it maintains a priority list (similar to IPL Priority) and provides the user with the capability of editing the order and saving it in NV memory. The BIOS must then install the controllers in the order the user specified. The three types of controllers mentioned previously each install in different ways. Support for ATA drives is installed by executing internal BIOS code. PnP BCV devices require two calls, one to their option ROM which describes their capabilities to the BIOS, and then one to their BCV which allows them to hook into INT 13h. Finally, Legacy cards with option ROMs install entirely when their option ROM is called. So the BIOS must have the ability to dynamically control each of these events.

The installation ordering is very similar to the process used for managing the IPL Priority (BAIDs and BEV devices). A table of controllers called the BCV Table is built early in POST which includes all three types of controllers and pointers to code that can install them. A set of ordinal values called the BCV Priority is stored in NV memory and is used to index a particular controller in the BCV Table. The number of BCV devices is also stored in NV memory and is checked at each POST to see if a change occurred. If so, the BCV Priority is adjusted automatically by the BIOS. A sample BCV Priority is depicted below.





When the devices in the BCV Table are installed, the BIOS will process the BCV Priority shown above in the following order: BCV #1, ATA Drives, Legacy Cards, BCV #2.

5.4 POST Pseudocode

The following pseudocode describes an example of managing the BCV Priority during POST.

Created at BIOS build time:

- First two BCV Table entries are filled in by ATA support and Legacy option ROM support.
- The NV memory space for the IPL Priority is reserved.

Assumptions:

- The default for the BCV Priority will automatically be created during POST in case the NV memory gets corrupted.

1. Call the video option ROM.
2. Initialize the first two BCV Table entries with ATA Support and Legacy Cards respectively. These two entries will always exist in the BCV Table and will always be in that order.
3. Identify all option ROMs with BCVs and add any that are found to the BCV Table.
4. Call all the BCV option ROMs in the order they reside in the BCV Table and store their return value in AX upon return.
5. If NV memory is corrupted, set defaults for BCV Priority and BCVCount.
6. Check if the number of BCV devices found this time matches the number found last time. If not, update the BCV Priority and BCVCount accordingly.
7. FOR (i = 0; i < BCVCount + 2; ++i)
   - index = BCV Priority[i].
   - Call BCVTable.InstallRoutine[index].
   - There are three possibilities:
     BIOS ATA support:
     - Call installation routine for on-board ATA.

     BCV Device:
     - Set up registers for BCV call.
     - Call the BCV.

     Legacy Cards:
     - Do Legacy ROM scan.
8. When INT 19h is invoked and INT 13h drive 80h is accessed for booting, the first BCV device that installed will respond and boot the O/S.

6.0 POST Sequence

6.1 Power-On Initialization

6.1.1 Initializing BAIDs

At some time during POST all BAIDs have to be initialized so that they are ready to boot. This would normally include floppy drives and hard drives, but may also include any other BAIDs such as CD-ROM, PCMCIA, embedded network adapters, etc. Since BAIDs are specific to the BIOS, it is the BIOS' responsibility to initialize all BAIDs.

6.1.2 PnP Boot Devices

During POST, the PnP BIOS may choose to configure some or all of the devices in the system. However, as per the Plug and Play BIOS Specification it is required that a PnP BIOS configure all IPL devices including BEVs and BCVs. Any of these devices could be on any bus or accessed over a bus bridge, so busses and bridges must be enabled as well. All option ROMs must be mapped into system memory between C0000h and EFFFFh.

6.2 PnP Option ROM Initialization

It is a requirement for all PnP option ROMs to return control back to the BIOS after they have completed their initialization. If a PnP option ROM does not return, it takes control away from the BIOS and foregoes compatibility with the BIOS Boot Specification.

First, the video option ROM is called, because it is a special case. Next, only the option ROMs containing a valid PnP Expansion Header with either a BEV or a BCV are called. The IPL Table and BCV Table are constructed after all option ROMs containing a PnP Expansion Header have been called. This allows an option ROM that operates under the DDIM (such as PCI) to add PnP Expansion Headers as devices are found. See Appendix E for more information on PCI devices with PnP Expansion Headers.

All PnP Cards have their option ROMs initialized by a far call to offset +03h in the option ROM header. Option ROMs in PnP Cards will be called in the order of lowest to highest with those located at the lowest memory address called first. It is recommended that PCI devices store their PFA passed in AX when their option ROM is called for initialization. When the far call is made to the PnP option ROM the following parameters will be passed:

ES:DI	=	Pointer to the PnP Installation Check Structure
AX	=	PFA - PCI Bus/Device/Function (PCI devices only)
BX	=	Card Select Number (PnP ISA devices only)
DX	=	Read Data Port Address (PnP ISA devices only)

During this initialization stage, none of the option ROMs called should hook any interrupt vectors relating to booting such as INT 9h, 10h, 13h, 18h, or 19h. After returning from a PnP option ROM initialization far call, AX will contain status information as to the device's booting capabilities. This word is saved for later use during the POST process. See the Plug and Play BIOS Specification for the bit definitions of AX after the return.

Option ROMs with a PnP Expansion Header will not be recognized if either one of the following is true:

- It is an option ROM utilizing the Device Driver Initialization Model (DDIM) and has re-sized itself to a size of zero.

- The PnP option ROM returns zero in AX after the initialization call and it has no BEV.

The rest of the PnP option ROMs that have not been called by now are considered to be Legacy because they did not contain a PnP Expansion Header.

6.3 Check IPL Priority and BCV Priority

This is the stage where the contents of NV storage are checked for corruption. If corruption is detected because the check sum failed (or for some other reason), a default IPL Priority and BCV Priority that represents the number of devices present must be constructed.

This is also where the number of BEV devices and BCV devices found during POST is compared against the number that was found at the last POST. If a change is detected, the new number of devices must be stored and the IPL Priority as well as BCV Priority must be updated to reflect the addition or loss of devices.

6.4 INT 13h Device Controller Installation

When an INT 13h device controller installs its INT 13h support, it typically snoops the BDA to detect if any hard drives are currently installed. If so, the controller will make its hard drive number one greater than the maximum drive number currently used. For example, if two ATA drives are installed (80h and 81h) and a BCV gets called to install support for a drive, the BCV will make its INT 13h drive number 82h. This precludes it from being an IPL device since only drive 80h is bootable. The result of all this is that the first INT 13h device controller that successfully installs will have its first drive assigned number 80h and will represent the Hard Drive BAID in the IPL Table and IPL Priority.

One of the keys to controlling the order of the installation of the INT 13h device controllers is selectively calling different option ROMs at specific times. It is critical that the BIOS allow the installation of controllers in any order. The three types of controllers found in the BCV Table are: those with on-board ATA support; those with Legacy option ROMs; and all of the BCV devices that were found. Each of these types of controllers has a different method for installing their INT 13h support. The ATA support is a call into the BIOS, the Legacy card support is also a call into the BIOS that simply calls the remaining option ROMs that don't have the PnP Expansion Header, and the BCV devices are controllers that each have a special entry point within their PnP option ROM.

6.4.1 Boot Connection Vectors

BCV devices are not part of the IPL Priority. The reason for this is twofold. First, after a BCV has been called, there is no way to know how many drives it has and if any are bootable. Second, whichever drive ends up as INT 13h drive 80h will become the BAID for the system's hard drive.

Each PnP option ROM that contains a BCV will have that vector called. When a BCV is called, it allows an option ROM to detect if its device is attached or not and conditionally hook INT 13h. If the option ROM decides not to install its support, it should not hook INT 13h, but it should perform any clean-up before returning to the BIOS. For instance, the BDA should remain unmodified if the device did not install.

A device's BCV in the PnP Expansion Header is valid if it is not zero and the BEV is zero, since these two vectors are mutually exclusive. When the BIOS makes a far call to the BCV the following parameters are passed:

ES:DI	=	Pointer to PnP Installation Check Structure
AX	=	Flags for interrupt(s) that may be hooked (PnP ISA devices only)
BX	=	Card Select Number (PnP ISA devices only)
DX	=	Read Data Port Address (PnP ISA devices only)

Note that for PCI cards, AX will not contain the PFA. PCI devices that need their PFA during execution of their BCV should store the PFA passed earlier when their option ROM's initialization vector was called.

6.4.2 Disconnect Vector

Originally, it was thought that the DV would be called by the BIOS if the device's BCV was called and subsequent booting from the device failed. However, it was later discovered that current PnP option ROMs are more well behaved by checking during the BCV call if their device will properly respond to INT 13h calls or not, and simply not hooking INT 13h if those calls would fail. Because of this, the DV is not called by a BIOS supporting the BIOS Boot Specification, nor is it necessary to have a valid DV in the PnP Expansion Header. The DV should be NULL and can't be used for other storage purposes.

6.4.3 Legacy ROM Scan

A scan for Legacy option ROMs in system memory from C0000h to EFFFFh on 2k boundaries is performed in the order of lowest to highest. Any option ROM with a PnP Expansion Header is ignored. When an option ROM is found its initialization vector is executed by a far call to offset +03h in the option ROM header. The contents of CPU registers passed to and returned from the option ROM call are ignored with the exception of PCI option ROMs. PCI option ROMs will be called with the PFA in the AX register as per the PCI Specification.

6.4.4 On-board ATA Support

The installation of the BIOS' own ATA drive support into the INT 13h services should be performed in the same way that BCV devices install their support, by recognizing the drives already installed and adding on drive numbers above those previously installed.

For example, if the BCV priority was set to the following: [BCV device #1, ATA support, Legacy cards, BCV device #2], and if a BCV was the first INT 13h device controller to install and hook into INT 13h to add support for a drive, it would occupy drive 80h. Then when the BIOS installs its support for an ATA drive, it should occupy drive 81h.

6.5 INT 19h Processing

By the time INT 19h gets invoked, all types of IPL devices have been identified and initialized, all INT 13h device controllers have installed, and the IPL Table contains references to all the available IPL devices. The first INT 13h device controller that installed successfully will occupy drive 80h and will be represented by the Hard Drive BAID in the IPL Table.

All that is left is to sequentially go through the IPL Priority, use each ordinal value as an index into an entry in the IPL Table, and attempt to boot from that device. Each entry in the IPL Table contains a pointer to a procedure that will attempt booting. The BIOS calls this boot handler address. The first IPL device that succeeds will load the O/S. If IPL an device fails to boot, the BIOS regains control and selects the next IPL device in the IPL Priority. This process continues until all IPL devices have failed. If all IPL devices fail to boot an error message is displayed and the booting process starts over again.

Implementation Specific Note:
It may be useful for the items in the IPL Table to be displayed to the user at the end of the INT 19h handler if all devices have failed to boot an O/S. This way the user can attempt to rectify the situation. Alternately, the user could be prompted to enter Setup if it were accessible at that time.

6.5.1 Booting from BAIDs

The BIOS attempts to boot from a BAID by calling a boot handler address that is located within the BIOS. The responsibility of loading the boot sector from the device and transferring control to it falls on the boot handler procedure. If the device fails to boot, it should invoke INT 18h, since this is now the boot recovery vector.

When the boot handler is called, the BIOS passes a pointer to the PnP Installation Check Structure in ES:DI. This is so that once the boot handler has successfully loaded the device's boot sector into memory at address 0000:7C00h, execution control can be transferred with the following register contents:

ES:DI = Pointer to PnP Installation Check Structure
DL = Drive number used for the INT 13h (00h, 80h, etc.)

6.5.2 Booting from BEVs

The BEV, which is found in the PnP Expansion Header, is for devices which do not have INT 13h support routines in their option ROMs and require their own proprietary method to load the O/S. Devices utilizing the BEV method are typically remote program load devices such as network cards.

The BEV device can be thought of as a generic way to boot from any type of device and allows for the future definition of other IPL devices. A PnP Card's BEV is valid if it is not zero and the BCV is zero, since these two vectors are mutually exclusive.

If a BEV device is selected for booting, its BEV will be called from within the INT 19h handler. The device will then attempt to load boot code into memory and execute the O/S. If the O/S load fails, the device must clean up any memory areas it modified and then invoke INT 18h to allow control to return to the system BIOS. Specifically, data areas such as the interrupt vector table, the BDA, and the Extended BIOS Data Area must remain unchanged.

6.6 INT 19h Pseudocode

The following pseudocode describes an implementation of the INT 19h process.

Assumptions:

- Maximum number of supported BAIDs and BEV devices = 8, indexed as 0..7 in the IPL Priority.

Execution at INT 19h time:

- IPLcount = current number of BAIDs and BEV devices at this boot.
- FOR (i = 0; i < IPLcount; ++i)
    - currentIPL = IPL Priority[i].
    - Use currentIPL to index the IPL Table entry.
    - Do a far call to the entry's boot handler or BEV, if successful we never return.
    - IF (control returns via RETF, or an INT 18h)
        - Clean up the stack if necessary.
    - ENDIF
- Execute an INT 18h instruction.

6.7 INT 18h Pseudocode

The original INT 18h handler jumped into a ROM-based BASIC interpreter. On compatible PCs, INT 18h typically displays an error message, waits for a key stroke, and then executes an IRET instruction. The BIOS Boot Specification redefines INT 18h as the recovery vector for failed boot attempts. The following pseudocode describes an implementation of the INT 18h process. Note that INT 18h does not, and cannot return to its caller because the stack has been reset.

Execution at INT 18h time:

- Reset stack.
- IF (all IPL devices have been attempted)
    - Print an error message that no O/S was found.
    - Wait for a key stroke.
    - Execute the INT 19h instruction.
- ELSE
    - Determine which IPL device failed to boot.
    - Jump to a label in the INT 19h handler to try the next IPL device.
- ENDIF

6.8 Notes on the POST Process

- The Plug and Play BIOS Specification says that if a Legacy IPL device's option ROM captures INT 18h or INT 19h, the BIOS should save this vector and then restore the original one put there by the BIOS. The BIOS Boot Specification deviates from this in that these vectors are not recaptured after each Legacy option ROM returns from initialization. That would be considered unsafe.
- If the system attempts to boot from a formatted non-bootable (data) floppy diskette, the INT 13h reads will be successful and control will be transferred to the resident boot sector on the diskette after it is loaded into memory. However, this boot sector code will determine that this diskette doesn't contain an O/S and it eventually may issue an INT 19h to attempt to start the booting process over. This could result in the INT 19h handler being executed all over again, thus creating an infinite loop.
- If the system attempts to boot from a non-bootable formatted (data) hard drive, when its boot sector is loaded and starts executing, it discovers that an O/S is not present. This code may issue an INT 18h. The INT 18h instruction is issued instead of INT 19h because the hard drive boot sector assumes that the floppy drive has already attempted to boot and failed. This behavior is left over from the IBM AT.
- If a Legacy IPL device fails to boot it may do one of several things:
    1. It may display an error message and never return control to the BIOS, thus hanging the system.
    2. If the device trapped INT 13h, it may invoke INT 19h or INT 18h to attempt to allow the BIOS to select another device to boot.
    3. If the device trapped INT 19h, it may invoke INT 18h.
- There is currently no defined standard for the usage of conventional memory by option ROMs during POST. However, it's recommended that options ROMs needing temporary storage when their initialization vector is called use RAM in the first 64k segment that is not currently defined for other purposes. Specifically, the memory range from address 0000:7C00h to 0000:FFFFh can be used, provided the stack is not located in this area.

Appendix A: Data Structures

A.1 IPL Table and BCV Table Entry Data Structure

Name	Offset	Size	Description
deviceType	00h	WORD	See definitions below.
statusFlags	02h	WORD	See bit definitions below.
bootHandler	04h	FAR PTR	Far pointer to address of boot handler.
descString	08h	FAR PTR	Far pointer to ASCIIZ description string.
expansion	0Ch	DWORD	Reserved for future expansion.

Both the IPL Table and the BCV Table consist of one or more of the above data structures arranged contiguously. The same data structure is used for both IPL Table and BCV Table entries.

deviceType: An identification number that describes what type of device this is.
00h = Reserved
01h = Floppy
02h = Hard disk
03h = CD-ROM
04h = PCMCIA
05h = USB device
06h = Embedded network
07h..7Fh = Reserved
80h = BEV device
81h..FEh = Reserved
FFh = Unknown

statusFlags: See table of bit definitions below.

bootHandler: For IPL devices, this points to a procedure that will attempt to load an O/S from this device. For BCV devices, this points to a procedure that will allow the device to attempt to install into INT 13h services.

descString: This points to an ASCIIZ string that describes this device to a user.

expansion: Reserved for future definition. Must be zero.

Bit definitions of the statusFlags field:

Bits	Field	Value	Description
3..0	Old Position	0..15	This entry's index in the table at the last boot. For updating the IPL or BCV Priority if individual device detection is done.
7..4	(Reserved)	0	Reserved for future use, must be zero.
8	Enabled	0..1	0 = Entry will be ignored for booting (IPL); entry will not be called for boot connection (BCV). 1 = Entry will be attempted for booting (IPL); entry will be called for boot connection (BCV).
9	Failed	0..1	0 = Has not been attempted for boot, or it is unknown if boot failure occurred (IPL); entry connected successfully (BCV). 1 = Failed boot attempt (IPL); failed connection attempt (BCV).
11..10	Media Present	0..3	0 = No bootable media present in the device. 1 = Unknown if bootable media present. 2 = Media present and appears bootable. 3 = Reserved for future use.
15..12	(Reserved)	0	Reserved for future use, must be zero.

A.2 PnP Option ROM Header

Offset	Size	Value	Description
00h	BYTE	55h	Signature byte 1.
01h	BYTE	AAh	Signature byte 2.
02h	BYTE	Varies	Option ROM length in 512-byte blocks.
03h	4 BYTES	Varies	Initialization entry point.
07h	17 BYTES	Varies	Reserved.
18h	WORD	Varies	Offset to PCI data structure.
1Ah	WORD	Varies	Offset to expansion header structure.

A.3 PnP Expansion Header

Offset	Size	Value	Description
00h	BYTE	'$'	Signature byte 1.
01h	BYTE	'P'	Signature byte 2.
02h	BYTE	'n'	Signature byte 3.
03h	BYTE	'P'	Signature byte 4.
04h	BYTE	01h	Structure revision.
05h	BYTE	Varies	Length (in 16 byte increments).
06h	WORD	Varies	Offset of next header (0000h if none).
08h	BYTE	00h	Reserved.
09h	BYTE	Varies	Checksum.
0Ah	DWORD	Varies	Device identifier.
0Eh	WORD	Varies	Pointer to manufacturer string (Optional).
10h	WORD	Varies	Pointer to product name string (Optional).
12h	3 BYTES	Varies	Device type code.
15h	BYTE	Varies	Device indicators.
16h	WORD	Varies	Boot Connection Vector (BCV), 0000h if none.
18h	WORD	Varies	Disconnect Vector (DV), 0000h if none.
1Ah	WORD	Varies	Bootstrap Entry Vector (BEV), 0000h if none.
1Ch	WORD	0000h	Reserved.
1Eh	WORD	Varies	Static resource information vector (0000h if none).

A.4 PCI Data Structure

Offset	Size	Value	Description
00h	BYTE	'P'	Signature byte 1.
01h	BYTE	'C'	Signature byte 2.
02h	BYTE	'I'	Signature byte 3.
03h	BYTE	'R'	Signature byte 4.
04h	WORD	Varies	Vendor Identification.
06h	WORD	Varies	Device Identification.
08h	WORD	Varies	Pointer to Vital Product Data.
0Ah	WORD	Varies	PCI Data Structure Length.
0Ch	BYTE	Varies	PCI Data Structure Revision.
0Dh	3 BYTES	Varies	Class Code.
10h	WORD	Varies	Image Length.
12h	WORD	Varies	Revision Level of Code/Data.
14h	BYTE	Varies	Code Type.
15h	BYTE	Varies	Indicator.
16h	WORD		Reserved.

Appendix B: Run-Time Functions (Optional)

It may be desirable for an O/S or application program to access the features of the BIOS Boot Specification during run-time. An application may want to query the number and type of IPL devices in the system, display an IPL device menu and allow the user to organize the IPL devices, or specify that the BIOS should attempt booting from a particular device on the next system restart.

What follows is a set of extensions to the Plug and Play run-time functions that implement the optional BIOS Boot Specification run-time services. Functions 60h through 6Fh are henceforth reserved for the BIOS Boot Specification. Refer to the Plug and Play BIOS Specification for the details of the calling conventions.

Run-Time Function Restrictions:

1. This API is specific to the data structures defined in Appendix A.
2. Any device that is placed in the IPL Table or BCV Table cannot have its option ROM removed from memory.
3. In an implementation of this API, all defined data structures and fields must be valid.
4. If function 60h returns SUCCESS, then the rest of the functions must be implemented. This is because function 60h is used by the caller to effectively test for the presence of the other functions.

Function 60h - Get Version and Installation Check

Synopsis:

```
int FAR (* entryPoint) (Function, Version);
int Function;                  // PnP BIOS function 60h.
unsigned int FAR *Version;     // Version of BIOS Boot Specification returned.
```

Description:

Optional. This function is used to check for the presence of a BIOS that is compliant with the BIOS Boot Specification and return the version number.

Version. Contains the version number in binary coded decimal (BCD) format. For example, if BX returns with 0100h, the version is 1.00. The major version is incremented when this API changes. The minor version is incremented when underlying functionality changes and this API remains the same.

Returns:

0 if successful - SUCCESS.

!0 if an error (Bit 7 set) or a warning occurred or no pending events - error code. The function return codes are described in Appendix C of the PnP BIOS Specification.

Function 61h - Get Device Count

Synopsis:

```
int FAR (* entryPoint) (Function, Switch, Count, MaxCount, StructSize);
int Function;                  // PnP BIOS function 61h
int Switch;                    // 0 = IPL relative, 1 = BCV relative.
```

*unsigned int FAR *Count;* *// Number of devices in the system.*
*unsigned int FAR *MaxCount;* *// Maximum number of devices supported.*
*unsigned int FAR *StructSize;* *// Size of a Table entry in bytes.*

Description:

Optional. This function is used to return the number of IPL or BCV devices currently present, the maximum number of IPL or BCV devices supported, and the size of an IPL or BCV Table entry.

Count. Contains the number of IPL or BCV devices present

MaxCount. Specifies the maximum number of IPL or BCV devices supported.

StructSize. Designates the size of an IPL or BCV Table entry in bytes.

See appendix A for the structure of an IPL or BCV Table entry.

Returns:

0 if successful - SUCCESS.

!0 if an error (Bit 7 set) or a warning occurred or no pending events - error code. The function return codes are described in Appendix C of the PnP BIOS Specification.

Function 62h - Get Priority and Table

Synopsis:

int FAR (entryPoint) (Function, Switch, Priority, Table);*
int Function; *// PnP BIOS function 62h.*
int Switch; *// 0 = IPL relative, 1 = BCV relative.*
*unsigned byte FAR *Priority;* *// Copy of Priority in NV memory returned.*
*unsigned byte FAR *Table;* *// Copy of Table.*

Description:

Function 61h Get Device Count must be called prior to calling this function so that the proper memory buffers can be allocated by the caller.

Optional. This function is used to copy the IPL or BCV Priority and IPL or BCV Table into the caller's destination buffers.

Priority. A byte array of size *maxCount*, the maximum number of IPL or BCV devices supported in the system. Only the first *Count* bytes in the IPL or BCV Priority are valid.

Table. An array of IPL Table or BCV Table entries containing information about each IPL or BCV device. The number of IPL or BCV Table entries is equal to *maxCount*. The size of the *Table* buffer is equal to *(maxCount * StructSize)* bytes. The caller must allocate enough memory for the destination buffers.

Returns:

0 if successful - SUCCESS.

!0 if an error (Bit 7 set) or a warning occurred or no pending events - error code. The function return codes are described in Appendix C of the PnP BIOS Specification.

Example of Boot Priority:

In a system where *maxCount* = 8, *Count* = 3; where there are two BAIDs, one BEV device; and the IPL Priority = 02h, 01h, 00h representing a boot order of network card, first hard drive, first floppy drive; *Priority* would point to a buffer containing the following:

02h	01h	00h						**IPL Priority.**

and *Table* would point to a buffer containing the following:

01h	00h	23h	E1h	00h	F0h	45h	E3h	00h	F0h	**first BAID, first floppy drive**
02h	00h	45h	E2h	00h	F0h	60h	E3h	00h	F0h	**second BAID, first hard drive**
80h	00h	E0h	0Eh	00h	D4h	48h	00h	00h	D4h	**first BEV, PnP ethernet card**

Function 63h - Set Priority

Synopsis:

int FAR (entryPoint) (Function, Switch, Priority);*
int Function; *// PnP BIOS function 63h.*
int Switch; *// 0 = IPL relative, 1 = BCV relative.*
*unsigned byte FAR *Priority;* *// Priority to write to NV memory.*

Description:

Function 62h Get Priority and Table must be called prior to calling this function so that the original *Priority* has been obtained. The contents of *Priority* are justified toward the beginning. The entries in *Priority* may be moved from one location to another, but it is required that no new values are written into *Priority*.

Optional. This function is used to write an IPL or BCV Priority into NV RAM.

Priority. A byte array of size *maxCount*, the maximum number of devices supported in the system. Only the first *Count* bytes in *Priority* are written into NV RAM.

Returns:

0 if successful - SUCCESS.

!0 if an error (Bit 7 set) or a warning occurred or no pending events - error code. The function return codes are described in Appendix C of the PnP BIOS Specification.

Function 64h - Get IPL Device from Last Boot

Synopsis:

int FAR (entryPoint) (Function, IPLEntry);*
int Function; *// PnP BIOS function 64h.*
*unsigned int FAR *IPLEntry;* *// Index of entry in IPL Table.*

Description:

Optional. This function is used to find out which device in the IPL Table was used on the last O/S boot.

IPLEntry. Contains an index of an entry in the IPL Table.

Returns:

0 if successful - SUCCESS.

!0 if an error (Bit 7 set) or a warning occurred or no pending events - error code. The function return codes are described in Appendix C of the PnP BIOS Specification.

Appendix C: Boot Menu (Optional)

C.1 Boot Menu Pop-up

Another way to view the IPL Priority is to use a Boot Menu that pops up during POST. If you press <Hot-Key> during POST, the Boot Menu will appear just before the BIOS issues INT 19h. The Boot Menu won't appear unless <Hot-Key> is pressed As in Setup, the Boot Menu displays all the IPL devices in the order they will be selected for booting. However, unlike in Setup, the Boot Menu doesn't allow you to edit the position of an IPL device in the IPL Priority. This menu simply allows you to select a Boot First device. A Boot First device attempts booting *first* before the regular IPL Priority is considered. You select a Boot First device from the Boot Menu by using the up/down arrow keys to move the highlight bar to the desired device, and then by pressing <Enter> the ordinal value from the IPL Priority for this selection is stored so that later the INT 19h handler can boot from this device. If you press <Hot-Key> while the Boot Menu is up, the Boot Menu exits without saving a selection.

The key defined as <Hot-Key> is dependent upon the particular implementation. For example, the <Esc> key could be used.

C.2 Boot Menu INT 19h Pseudocode

When the BIOS' INT 19h handler gets control, it checks to see if a Boot Menu IPL device was selected for booting. The following pseudocode is done before the normal INT 19h processing, which works only from the IPL Priority.

- IF (A Boot Menu selection was made)
    - currentIPL = IPL Priority[Boot Menu selection].
    - Use currentIPL to select the BAID or BEV table entry.
    - Do a far call to the boot handler, if successful we never return.
    - IF (we get control back via RETF, or an INT 18h):
        - Clean up the stack if necessary.
    - ENDIF
- ENDIF

C.3 Boot First Run-Time Functions

If it is necessary to duplicate the Boot Menu functionality in software, getting and setting the Boot First device could be accomplished by run-time functions instead of requiring the user to access a Boot Menu. The following two run-time functions could be used by an O/S or application program to get and set the Boot First selection.

Function 65h - Get Boot First

Synopsis:

int FAR (entryPoint) (Function, IPLEntry);*
int Function; *// PnP BIOS function 65h.*
*unsigned int FAR *IPLEntry;* *// Index of entry in IPL Table.*

Description:

Optional. This function is used to find out which device in the IPL Table is currently selected as the Boot First device.

IPLEntry. Contains an index of an entry in the IPL Table currently selected as the Boot First device. Valid values are 00h through FEh. A value of FFh means that there is currently no Boot First device selected.

Returns:

0 if successful.

!0 if an error (Bit 7 set) or a warning occurred or no pending events - error code. The function return codes are described in Appendix C of the PnP BIOS Specification.

Function 66h - Set Boot First

Synopsis:

int FAR (entryPoint) (Function, IPLEntry);*
int Function; *// PnP BIOS function 66h.*
*unsigned int FAR *IPLEntry;* *// Index of entry in IPL Table.*

Description:

Optional. This function is used to set the Boot First device. The Boot First device is used to boot from first before the IPL Priority is considered.

IPLEntry. Contains an index of an entry in the IPL Table to be the new Boot First device.

Returns:

0 if successful.

!0 if an error (Bit 7 set) or a warning occurred or no pending events - error code. The function return codes are described in Appendix C of the PnP BIOS Specification.

Appendix D: Recommended Boot Sector Changes (Optional)

If O/S's responded to the mechanism of passing the INT 13h drive number to the boot sector as defined by the Plug and Play BIOS Specification, the BIOS could boot from *any* INT 13h drive. Also, if a standard method of returning control to the BIOS upon boot failure were established, the BIOS could try to boot from the next device. Here are two recommended changes to the O/S boot sector code in order to enhance the booting capabilities of the BIOS.

D.1 Use DL for Drive Number

Use the drive number passed in the DL register by the BIOS when control is transferred to the boot sector for INT 13h accesses to load the O/S, instead of having the drive number hard-coded. This would allow booting from drives other than just 00h (A:) and 80h (C:).

D.2 INT 18h on Boot Failure

If an O/S is either not present, or otherwise not able to load, execute an INT 18h instruction so that control can be returned to the BIOS. Currently, hard drive boot sectors do this, but floppy diskette boot sectors execute an INT 19h instead of INT 18h. The BIOS Boot Specification defines INT 18h as the recovery vector for failed boot attempts.

Both of these solutions should be backward compatible with previous BIOS and O/S versions.

Appendix E: PCI with Multiple PnP Headers (Optional)

E.1 Description

The following procedure describes a method by which the option ROM on a PCI SCSI controller can allow a PC system BIOS to selectively recognize and install individual SCSI drives into the BIOS' INT 13h services. This is presented as an example of the more general case where a PCI option ROM with a PnP Expansion Header can implement multiple headers to allow recognition of individual devices.

The procedures described below allow the user to control the order in which individual drives are installed into the BIOS' INT 13h services. This has two major benefits. First, the order the drives appear in the INT 13h services controls the assignment of drive letters under the DOS and Windows environments. Although drives with multiple partitions can defeat this feature, in the worst case it still offers the user some control. Second, the first device to successfully install into INT 13h services will assign itself drive number 80h. Because of current operating system boot sector limitations, this is the only drive number that can be a boot device. Thus the user is selecting their boot device from any number of possibilities.

E.2 Requirements

- The computer must have a PCI bus, and the system BIOS must provide both PCI and Plug and Play support.
- The SCSI controller must be PCI and expect its option ROM to be shadowed.
- The system BIOS must support boot device control as defined earlier in the BIOS Boot Specification.

E.3 Option ROM Initialization

E.3.1 Before Option ROM Placement

One PnP Expansion Header structure exists in the PCI SCSI option ROM in the original binary image before the BIOS performs placement. The Boot Connection Vector (BCV) within this single header should be valid. At this point the header itself is merely for identification purposes so that the BIOS can enumerate all such devices in the system.

E.3.2 Placing the PCI Option ROM

The location of the option ROM in upper memory is chosen by the system BIOS. The segment address of the option ROM is determined by the value in the CS register when the ROM's initialization code gains control. The option ROM is shadowed into an available location within physical memory addresses C000h-EFC0h on a 2k boundary.

E.3.3 Calling the PCI Option ROM

It is guaranteed that all option ROMs will be called to initialize before the BIOS scans memory for option ROMs containing PnP Expansion Headers and builds the IPL and BCV Tables. This allows PnP Expansion Headers to be added and removed during their option ROM initialization without confusing the BIOS.

The option ROM has been shadowed and is write-enabled.

The CPU registers are set up as follows:

AX	PFA for this PCI adapter.
CS	Segment address of option ROM.
ES:DI	Far pointer to the $PnP Installation Check Structure in the system BIOS.

At this time, the option ROM performs any necessary initialization and determines the number of devices it will control. The option ROM may also determine if another adapter in the system has already installed and taken over its devices.

E.3.4 No Devices Present

If no devices are present, the option ROM should return to the BIOS without modifying the system interrupt vectors or other data areas. The return value in AX would be 0100h. This indicates that, although the adapter supports the INT 13h block device format, it failed to install properly because it had no devices attached.

E.3.5 Devices are Present

If one or more devices are present, the option ROM creates additional PnP Expansion Headers, one for each device found, in a linked list to the first one. Each header, including the first header, then represents an individual device. For example, if two SCSI drives are attached, header 1 (pointed to by offset 1Ah in the option ROM) represents the first drive (SCSI id 0), and the 'Offset of Next Header' field within header 1 is a link to header 2. Header 2 then represents the second SCSI drive (SCSI id 1) attached to the adapter. This can continue up to the maximum number of SCSI devices supported on that adapter. This linked list of headers terminates when a header's 'Offset of Next Header' field is zero.

The data within each header must be valid. Especially the 'BCV' and 'Pointer to Product Name String' fields. The BCV should point to a procedure that installs *only* that device into INT 13h services. It is strongly recommended that the Product Name String for each header uniquely identify the device to which that header belongs, so that when these strings are displayed to the user in a menu, the user can intelligently recognize and choose devices connected to that controller without having to open up the computer.

Multiple SCSI adapters can be accommodated by one SCSI option ROM using the scheme described above. It is only important that each header correspond to an individual device. For example, if two PCI SCSI adapter cards are present in a system, the option ROM for the first one called by the system BIOS could enumerate all the devices present on *both* adapters and create a linked list of PnP Expansion Headers covering all these devices accordingly. When the second adapter's option ROM is called by the system BIOS, it would determine that the first adapter is already present and controlling the second adapter's devices. The second adapter would then minimize the size of its option ROM to free UMB space, zero out the BCV within its PnP Expansion Header (so it won't be called), and then return to the system BIOS with AX containing 0100h indicating no devices were found.

E.4 Enumerating PnP Expansion Headers

Once all option ROMs with PnP Expansion Headers have been initialized, the BIOS will scan memory again and build a table of all the PnP Expansion Headers that are found so that the user can order these devices on a menu in Setup. The system BIOS will manage changes as devices are inserted and removed from the system as described earlier.

E.5 Calling the BCVs

The BIOS maintains the order in which the PnP Expansion Headers are called for installation. The calling of BCVs is driven from this order. Please refer to section 5.2.5 Controller Installation Guidelines for details on the expected behavior of the option ROM when its BCV is called. Essentially, each BCV procedure will install only its corresponding device into INT 13h services.

As outlined earlier, the system BIOS controls the installation order of all types of INT 13h devices via the BCV Table and BCV Priority. This includes ATA (IDE) drives, PnP BCVs, and a ROM scan for Legacy and non-PnP PCI cards.